Documentation of MINOS Detector Control Systems: Design and Implementation

E. Beall

September 2004

1 DCS Overview

The MINOS Detector Control Systems (hereafter referred to as DCS) covers all aspects of the detector state related to the running of the electronics that take the data, not the Data Acquisition System's (DAQ) state, nor the physics data itself. The DCS has been designed to provide information about the detector through:

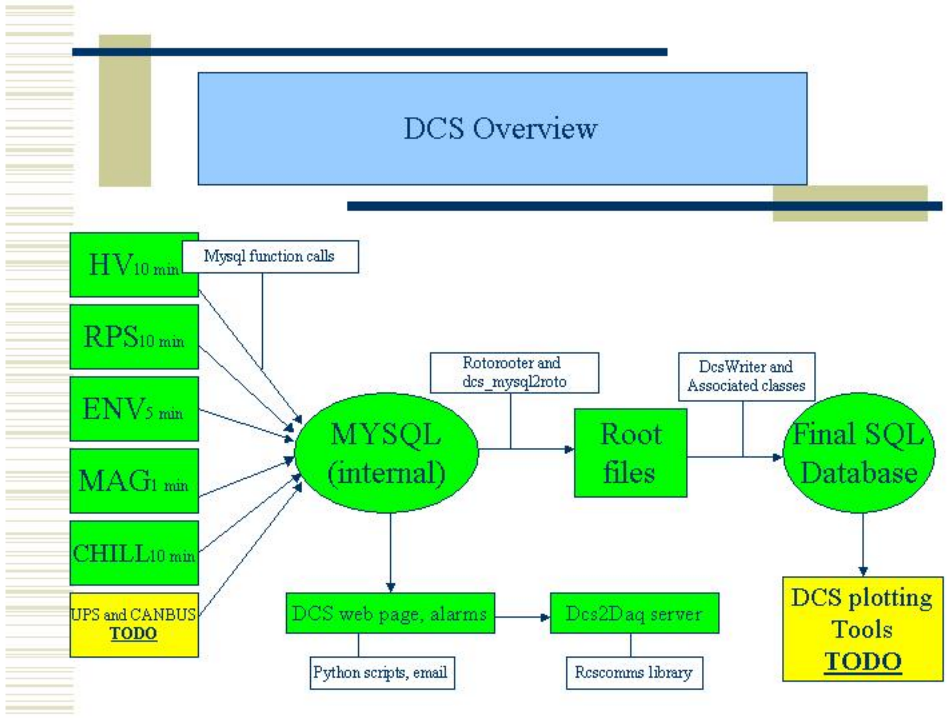
- 1: The DCS webpage, for online operations
- 2: A link to the DAQ server providing it with the global DCS state
- 3: Alarming and resetting some subsystems
- 4: The offline software reconstruction through the offline SQL database

To accomplish this, each subsystem is monitored by its own daemon and the state data is saved to an internal MySQL database. State data is logged upon state changes (if any) and at regular intervals from one minute to 100 minutes depending on the subsystem.

This internal database is used for all online operations, which are tasks 1-3 in the above list. The data is written to binary root files for archiving at fermilab's mass storage system and finally written back to the offline database in the format compatible with the offline database interface package (task 4).

The DCS tasks are separated into Online and Offline. The Offline should never affect the Online and vice versus, so long as each subsystem is successfully writing its data to the internal database. Some tasks are handled by python scripts. In the future, all but the environment and magnet monitors may be re-written in python.

The image below describes the overall scheme:



2 Basic: DCS subsystems

The Far Detector requires High Voltage (HV), Rack Protection System (RPS), Environment (ENV), Magnet (MAG and MAG2), Chiller (CHILL), CAN-Bus (CAN) and UPS (UPS) monitoring. The labels I have given in the previous sentence for each system are the prefixes of each table in the internal MySQL database. There are two tables for Magnet due to each supermodule requiring its own coil and supply.

The Near Detector requires High Voltage (HV), Rack Protection System (RPS), Environment (ENV), CAN-Bus (CAN) and UPS (UPS). Magnet and Chiller (the NearDet's chiller is used for cooling all the crate's electronics, not the magnet coil) will be provided by ACNET and it will not need its own table as yet.

The Calibration Detector, now decomissioned, used Environment (ENV) and Magnet (MAG). The Environment also collected environment data as well as beam spills, while Magnet collected state information about focussing and bending magnets in the beam line. Data rates were very high and put a great strain on the MySQL server, but accomplished this purpose well. For this reason it was not recommended to use DCS for beam monitoring, and ultimately a separate system was setup to handle this which is entirely separate from DCS.

2.1 HV daemon

The HV monitor controls the LeCroy HV mainframes at all detectors, and a single instance of the program must be run at each detector, and two are required for the FarDet due to the two supermodules. The program was written in c++ on linux. It can use the serial port or the EDAS ethernet to serial converter (with the associated drivers that must be compiled in). It is a complicated program and will provide the functionality we need for the two detectors, but if any new functionality is requested, a serious rewrite of the code is advised. This program uses MySQL function calls described later, so if that code is ever recompiled, this program must be recompiled as well. The hv monitor is run as a bash script that actually calls the executable, see the DCS user manual for details. The bash script calls hv.exe every 5 minutes per supermodule, and so it writes all phototube's voltages to database every 5 minutes. The FarDet's hv monitors are run from dcshv1.minos-soudan.org and dcshv2.minos-soudan.org and reside in the hvsrc subdir on both computers. The NearDet hv monitor runs on dcsdcp-nd.fnal.gov, in the hvsrc subdir.

The rate of data written has been problematic for the MySQL server since data queries of this table can fail if the query requires temp tables to be made and tehse temp tables reach the limit of disk size. A major redesign of the various subsystems is intended to remove this problem, although at the moment, the work-around has been to allocate more disk space perform less automated checking of these largest tables.

2.2 RPS daemon

The RPS daemon monitors the Rack Protection System modules that are located at the top of each rack and proved basic power, water and fire protection (by shutting-off during an alarm) to the electronics. It is written in c on linux and uses the netlink libraries (another ethernet to serial port controller software driver), which can be retreived from Alec Habig's cvs repository (contact him for access). The code resides on dcsdcp.minos-soudan.org at FarDet in the rps subdir, and on dcsdcp-nd.fnal.gov at NearDet also in the rps subdir. This code also uses the MySQL libraries I will describe later for writing to database, so anytime that code is recompiled, this must also be recompiled. It writes the status of every RPS to database upon initial run of the program, every 100 minutes, and also upon any state change.

2.3 ENV daemon

This daemon was written in visual basic on Windows XP (originally Win2K), and requires visual basic MySQL drivers found at www.icarz.com/mysql. The daemon polls National Instruments FieldPoint ethernet devices every couple seconds, and every five minutes at FarDet and NearDet, it writes the data to database. The code is quite simple. At NearDet, if the temperature of any rack gets above 90 degrees, it will be written immediately since this temperature is used as the warning for the CAN-Bus shutdown utility. More about this later.

2.4 MAG daemon

Exists only at FarDet. This daemon was written just like the environment monitor, only it sends its data to database every minute, for each supermodule. A different program (slightly different hard-coded values) is run for each supermodule. This DCS program is very important in that it can control the current running through each supermodule's coil. Eventually, a degauss cycle will need to be coded into it by the B-Dot group. The data rate from this subsystem is a concern as well.

2.5 CHILL daemon

Exists only at FarDet. This is a python script that connects to a ethernet to serial converter that presents a telnet interface to the chiller data. This is a little box beside the first floor of the elevator, on the west side. The ethernet box is strung by serial port to the actual chiller controller box, which is on the wall on the west side of the elevator. I mention this because during development, it was necessary to power cycle the serial to ethernet converter, because it is what runs the simple telnet server and would sometimes get confused during development. Dave Saranen is the contact person for questions about this hardware, if it ever becomes an issue.

2.6 CAN daemon

This will record the voltages and currents present on the VME crates. The daemon to poll the CAN-BUS devices consists of a server daemon which does the polling, and a python script to interface with this daemon to shutdown individual crates at the NearDet if the environment finds their temperature rising above 90 degrees F. At NearDet, the shutdown script looks in the internal database to see what the temperatures of the crates are, and if any are above 90 degrees, it interfaces with the server to shut them down individually.

2.7 UPS daemon

Not yet implemented. Will one day perhaps interface with UPS's that are now installed at FarDet on all crates, and possibly the coil, to record status.

3 Pre-Online: writing data

MySQL function calls have been written into a c-library for the linux-based subsystems residing in ~minos/DcsDatabase/EriksDcs/databasedrivers. Makefile and all, it can be recompiled and installed with "make install" in ~minos/external libraries. For the visual basic monitors, function calls to MySQL are available from ”www.icarz.com/mysql” and have been installed on the windows DCS computers.

3.1 Internal data writing scheme

The internal MySQL database has a table for each subsystem (two for MAG, or rather MAG and MAG2 at FarDet), with the prefix for the table prefix followed by TABLE (like HVTABLE or MAG2TABLE). Each table has a column for each data member, and one column for SEQNO and one for the timestamp, with the subsystem prefix (like HV_TIMESTAMP, although MAG2 uses just MAG for prefix to this column). The SEQNO links this table with an additional VLD table (like HVTABLEVLD) that contains SEQNO and ROTO columns. The other columns are mostly unimportant, with the exception of the TIMESTART column of HVTABLEVLD spanning one HV record, while the HV_TIMESTAMP's in the main data table could span several minutes.

For our purposes, it is very important that SEQNO for each block of data is incremented every write. Beyond that, the tables must only be synchronized by this SEQNO. For each record block, ROTO is written as 0. One record block can consist of many rows (HV) linked via SEQNO to a single VLD entry. When the data is read out from this and put into root files later, the ROTO column is modified to 1, marking the record as having been stored.

4 Online: DCS web pages

The web pages are generated by python scripts that access the internal database using MySQL-db python package. Two scripts exist in the dcs_scripts subdir. The shell script dcsenv.sh calls the python script dcsweb.py, so the shell script can be called from crontab and copy files using kerberized scp to the web machine so they can be viewed globally.

Apart from presenting the current information to the world, this web page maker also writes the entry regarding global dcs state for the dcs2daqServer to read. This is written in the database and reflects whether the problem (if any) is the magnet coil, an RPS or high voltage and as such reflects the quality status of the data being taken. This status is relayed to the DAQ process for further data quality monitoring.

5 Online: DCS2DAQ communication

This is a multi-threaded c-program that queries the internal database for the table DCS_GLOBAL_STATE every minute, and presents this state on another thread to a port the DAQ has been configured to watch. It uses RcsComms library from the “online” cvs repository (Tim Nicholls is contact person). It makes the DCS status button go red or green on the DAQ web page and DAQ gui depending on connection status and DCS status. For this purpose, it is assumed that a Magnet coil off, RPS down, or HV mainframe tripped is an error state.

6 Offline: making root files

This step requires use of the offline software packages DcsDaemon and RotoRooter. RotoRooter needed no modifications for DCS, but DcsDaemon is the package containing a single source file dcs_mysql2roto that queries each internal table in order to see if new data has been written. This is the reason for the ROTO column in each subsystem's VLD table to tell if a row of data has yet been written to root file. Refer to the section in the offline software user's manual about RawData and the RotoRooter. Each raw block is labelled by a Block Identifier (defined in OnlineUtil/rawBlockIds.h), the detector type, data type and a minor version number. This is because the block's evolved over time, and have a fundamentally different structure for the three different detectors. The block is an array of ints, so care must be taken to keep floats in machine independent format.

6.1 usage of dcs_mysql2roto

At FarDet, on dcsdcp, go into subdir dcsRoto. There is a script in /etc/init.d/rotoroot that shows how it must be started. rotorooter must be started as "rotorooter -e -p dcs", listening to the dcs port, and then the daemon may be run as "dcs_mysql2roto -d far". An environment variable must also be set for DCS_DATA_DIR, currently set to /dcsdata. It closes its current root file and starts a new one every 12 hours.

6.2 archiving

Any closed root files are archived by Liz Buckley-Geer's archiver by usage of a bash script that looks for new *.mdcs.root files in this directory that are not currently open. The bash script (archtouch) touches the filename of the file to be archived in /dcsdata/archiver/data-to-archive. When it is archived, the touch is removed, and a filename touch is placed in /dcsdata/archiver/data-archived subdir. This bash script is called once every 12 hours.

6.3 data disks

The disk that the root files resides on is intentionally different from the one that holds the internal MySQL database to ensure some redundancy. For example, /dcsdata holds the data root files, and is mounted on the main hard drive, while the sql database dir is mounted on a second hard drive as one partition, and a backup dir is mounted on this second drive as a second partition.

6.4 cronjobs and other notes

Some cronjobs run checkscripts that just ensure things are running, and send email if they're not running. Other cronjobs do much more important things. Type crontab -l to see the crontab. "archtouch" sets up the touched files so that the archiver will send the root files to fermilab. The "mysqlcheck" bash script checks if the MySQL server is running, "rpscheck" checks for rpsdaemon, "rotocheck" checks for rotorooter and "dcs_mysql2roto" and "dcsServeCheck" checks for the dcs2daqServer. Finally, "ChillerEnv.sh" is the chiller monitor.

"filefinder_cronjob.sh" is the script that looks for new root files and writes them to the offline database. "dcsenv.sh" is the web page maker and overall DCS master monitor. This runs every two minutes at FarDet and every one minute at NearDet. The only caveat with this script is that the MySQL queries take a lot of time. The data in the internal database must be cleaned out every month or two to allow this script to run within a minute. So therefore there is this other script "eriks_sqlbackup" that cleans out anything up to the last month of data from the sql database and optimizes tables. This script temporarily disables the web page to remove load on the MySQL database server while it works. Runs once monthly and requires between ten and fifteen minutes to run. It displays a maintenance message on the DCS web page.

The subdirectories in user minos's home dir are important. The minossoft dir is the local installation of minossoft. The external dir contains all libraries needed for minossoft (root and MySQL as well as the MySQL data dirs, which are mounted on a separate disk in this dir) as well as the DCS-MySQL function call library discussed above under Pre-Online, writing data. The bin dir contains binaries called from scripts by user minos, such as rpsdaemon and the archiver.

7 Offline: reading the root files

Reading out a raw block requires RawData package. Needs the same care put into each block's methods as dcs_mysql2roto's blocks. For example, for the RPS subsystem, the dcs_mysql2roto job makes blocks with "kMdBlockDcsRpsMonitor" as the header (see OnlineUtil/rawBlockIds.h) and each block must be unpacked carefully. See RawData/RawDcsRpsMonitorBlock.h and .cxx and in particular notice the number of bytes (times 4 since these are stored as long ints) I add to each block and see the constructor in .cxx for doc on what each unit means. Same goes for all other blocks, except that each detector and minor version id (very important, when you add functionality that changes the size or format of the block, you must increment the minor version id or) sometimes the block formats are different. A simple loon (spin_rawdcs.C) script to read out a raw block root file (denoted by *.mdcs.root) exists in DcsUser/scripts. And look in the text output to see that you've gotten the blocks right. Validate it by comparing the output of this with the internal database at NearDet and FarDet, depending on which detector and root file, and remembering to account for the fact that the internal db runs on CST and the root file's timestamps are UTC.

8 Offline: Writing to offline database

Uses the RawData classes above and DcsUser class to write to database. Software for writing exists in DcsUser/DcsWriter.cxx, and scripts that seek out DCS root files exist in scripts/filefinder_cronjob$det.sh, where $det is either near or far. The scripts each contain several hard-coded values for NearDet and FarDet, notably the offline database passwords and url.

References

[1] E. Beall: *DCS Users Guide* http://webusers.physics.umn.edu/∼ebeall/dcscontrol.html August 2004

[2] The MINOS collaboration, *The MINOS Detectors Technical Design Report* October 1998

[3] *The MINOS Off-line Software User's Manual* http://www-numi.fnal.gov/offline_software/srt_public_context/doc/UserManual/UserManual.html September 2004